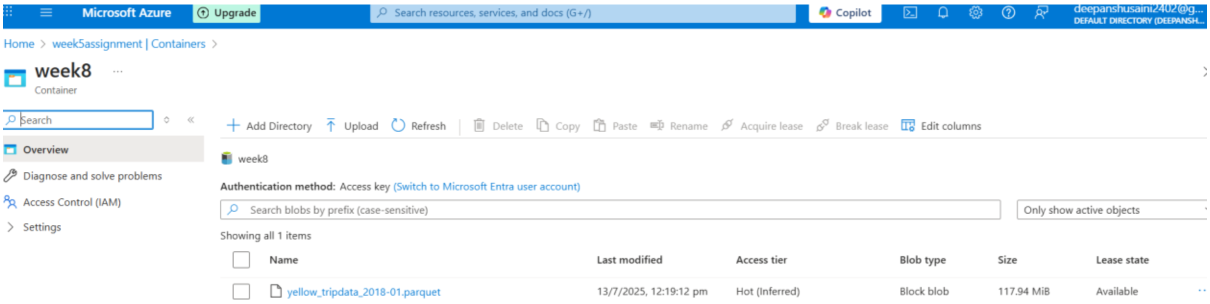
CELEBAL TECHNOLOGIES PVT. LTD.

ASSIGNMENT

WEEK 8

Load NYC taxi data to DataLake/Blob_Storage/DataBricks



week8

Container

week8

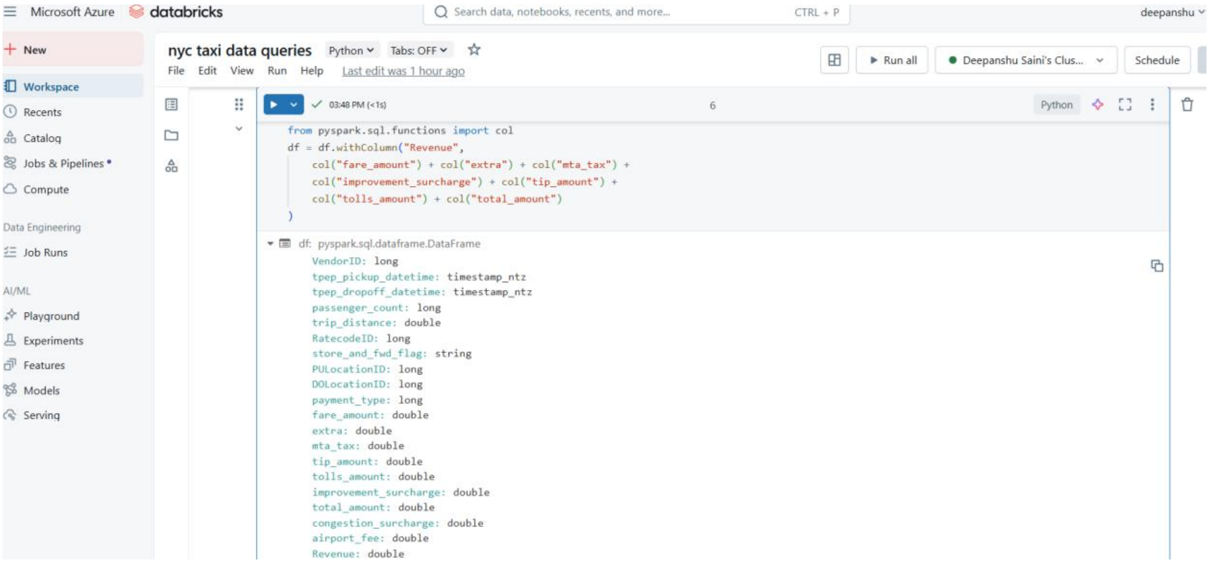
Authentication method: Access key (Switch to Microsoft Entra user account)

Only show active objects

Showing all 1 items

Name	Last modified	Access tier	Blob type	Size	Lease state
yellow_tripdata_2018-01.parquet	13/7/2025, 12:19:12 pm	Hot (Inferred)	Block blob	117.94 MiB	Available

Query 1. - Add a column named as ""Revenue"" into dataframe which is the sum of the below columns: 'Fare_amount', 'Extra', 'MTA_tax', 'Improvement_surcharge', 'Tip_amount', 'Tolls_amount', 'Total_amount'

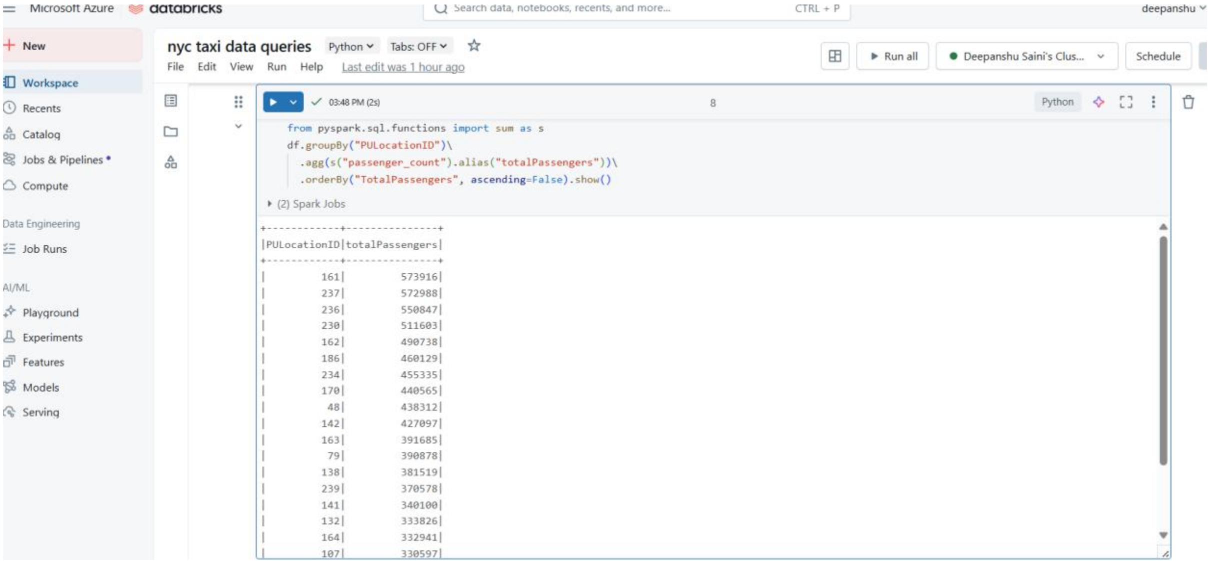


nyc taxi data queries

```
from pyspark.sql.functions import col
df = df.withColumn("Revenue",
    col("fare_amount") + col("extra") + col("mta_tax") +
    col("improvement_surcharge") + col("tip_amount") +
    col("tolls_amount") + col("total_amount")
)
```

```
df: pyspark.sql.dataframe.DataFrame
    VendorID: long
    tpep_pickup_datetime: timestamp_ntz
    tpep_dropoff_datetime: timestamp_ntz
    passenger_count: long
    trip_distance: double
    RatecodeID: long
    store_and_fwd_flag: string
    PULocationID: long
    DOLocationID: long
    payment_type: long
    fare_amount: double
    extra: double
    mta_tax: double
    tip_amount: double
    tolls_amount: double
    improvement_surcharge: double
    total_amount: double
    congestion_surcharge: double
    airport_fee: double
    Revenue: double
```

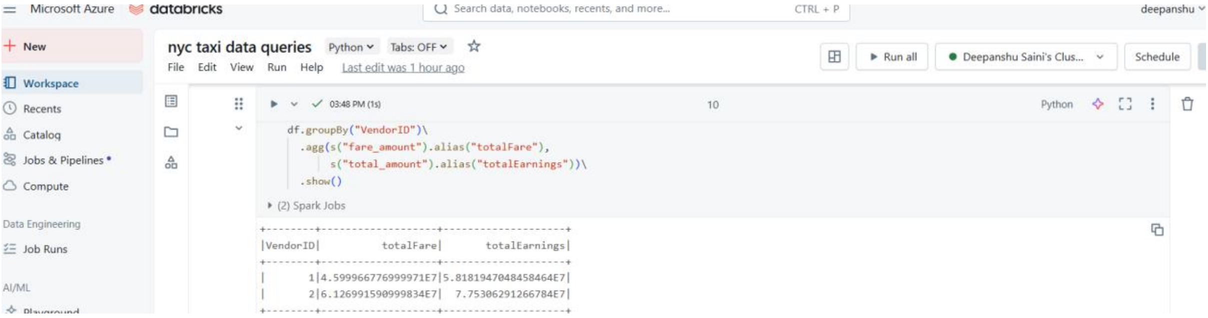
Query 2. - Increasing count of total passengers in New York City by area



```
from pyspark.sql.functions import sum as s
df.groupBy("PULocationID")\
  .agg(s("passenger_count").alias("totalPassengers"))\
  .orderBy("TotalPassengers", ascending=False).show()
```

```
+------------+---------------+
|PULocationID|totalPassengers|
+------------+---------------+
|         161|         573916|
|         237|         572988|
|         236|         550847|
|         230|         511603|
|         162|         490738|
|         186|         460129|
|         234|         455335|
|         170|         440565|
|          48|         438312|
|         142|         427097|
|         163|         391685|
|          79|         390878|
|         138|         381519|
|         239|         370578|
|         141|         340100|
|         132|         333826|
|         164|         332941|
|         107|         330597|
```

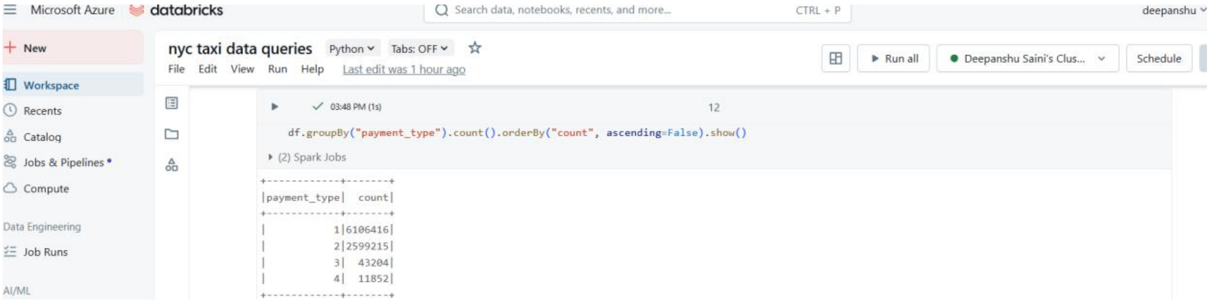
Query 3. - Realtime Average fare/total earning amount earned by 2 vendors



```
df.groupBy("VendorID")\
  .agg(s("fare_amount").alias("totalFare"),
       s("total_amount").alias("totalEarnings"))\
  .show()
```

```
+--------+------------------+------------------+
|VendorID|         totalFare|     totalEarnings|
+--------+------------------+------------------+
|       1|4.599966776999971E7|5.8181947048458464E7|
|       2|6.126991590999834E7|  7.75306291266784E7|
+--------+------------------+------------------+
```

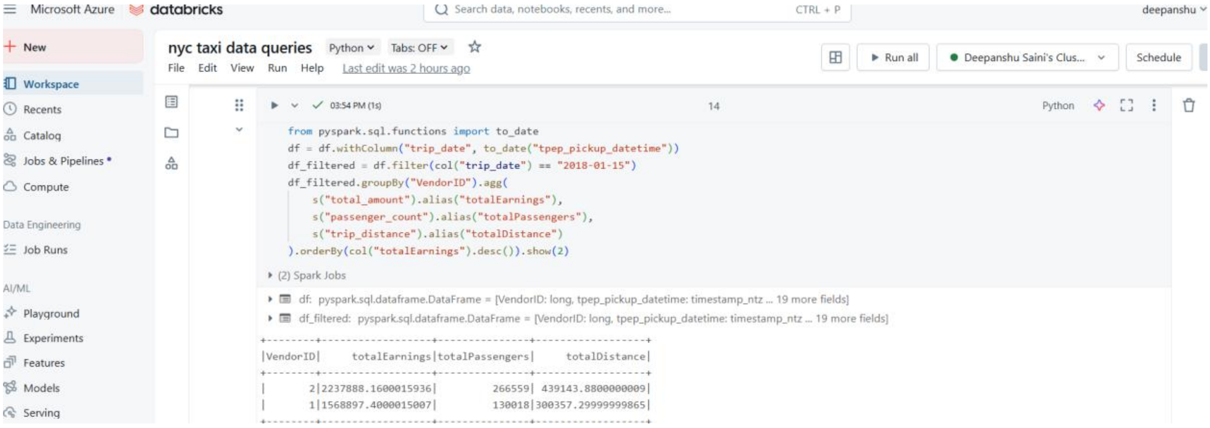
Query 4. - Moving Count of payments made by each payment mode



```
df.groupBy("payment_type").count().orderBy("count", ascending=False).show()
```

```
+------------+-------+
|payment_type|  count|
+------------+-------+
|           1|6106416|
|           2|2599215|
|           3|  43204|
|           4|  11852|
+------------+-------+
```

Query 5. - Highest two gaining vendor's on a particular date with no of passenger and total distance by cab

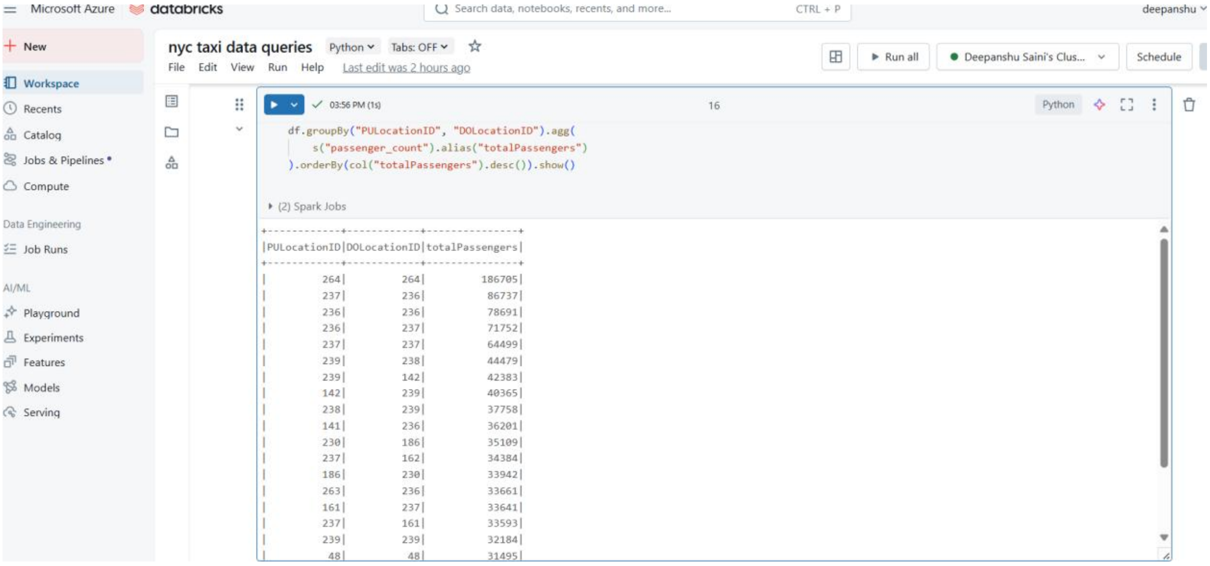


```
from pyspark.sql.functions import to_date
df = df.withColumn("trip_date", to_date("tpep_pickup_datetime"))
df_filtered = df.filter(col("trip_date") == "2018-01-15")
df_filtered.groupBy("VendorID").agg(
    s("total_amount").alias("totalEarnings"),
    s("passenger_count").alias("totalPassengers"),
    s("trip_distance").alias("totalDistance")
).orderBy(col("totalEarnings").desc()).show(2)
```

```
▸ (2) Spark Jobs
▸ df:  pyspark.sql.dataframe.DataFrame = [VendorID: long, tpep_pickup_datetime: timestamp_ntz ... 19 more fields]
▸ df_filtered:  pyspark.sql.dataframe.DataFrame = [VendorID: long, tpep_pickup_datetime: timestamp_ntz ... 19 more fields]

+--------+------------------+---------------+------------------+
|VendorID|     totalEarnings|totalPassengers|     totalDistance|
+--------+------------------+---------------+------------------+
|       2|2237888.1600015936|         266559| 439143.8800000009|
|       1|1568897.4000015007|         130018|300357.29999999865|
+--------+------------------+---------------+------------------+
```

Query 6. - Most no of passenger between a route of two location.

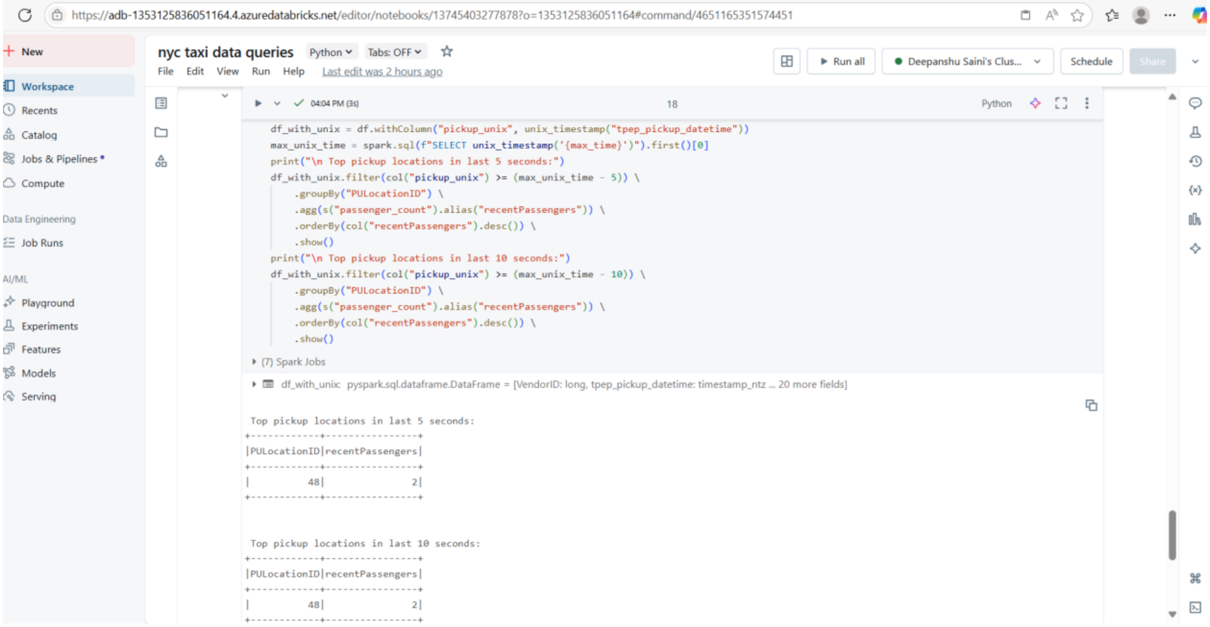


```
df.groupBy("PULocationID", "DOLocationID").agg(
    s("passenger_count").alias("totalPassengers")
).orderBy(col("totalPassengers").desc()).show()
```

```
▸ (2) Spark Jobs

+------------+------------+---------------+
|PULocationID|DOLocationID|totalPassengers|
+------------+------------+---------------+
|         264|         264|         186705|
|         237|         236|          86737|
|         236|         236|          78691|
|         236|         237|          71752|
|         237|         237|          64499|
|         239|         238|          44479|
|         239|         142|          42383|
|         142|         239|          40365|
|         238|         239|          37758|
|         141|         236|          36201|
|         230|         186|          35109|
|         237|         162|          34384|
|         186|         230|          33942|
|         263|         236|          33661|
|         161|         237|          33641|
|         237|         161|          33593|
|         239|         239|          32184|
|          48|          48|          31495|
```

Query 7. - Get top pickup locations with most passengers in last 5/10 seconds.



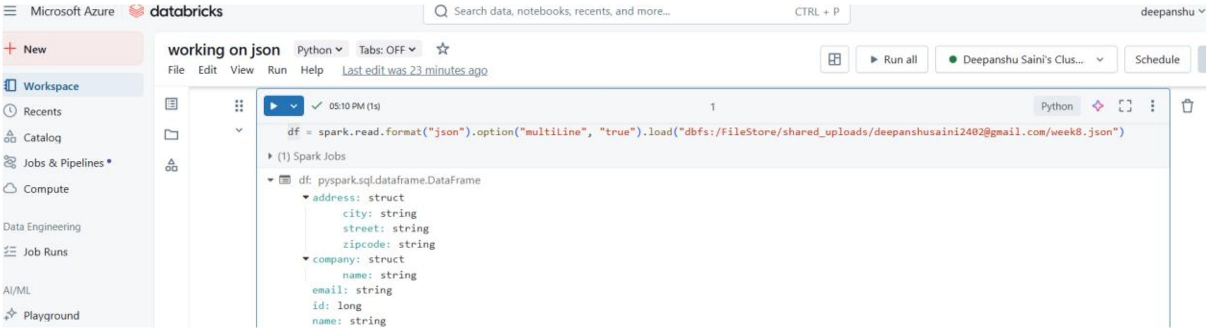
```
df_with_unix = df.withColumn("pickup_unix", unix_timestamp("tpep_pickup_datetime"))
max_unix_time = spark.sql(f"SELECT unix_timestamp('{max_time}')").first()[0]
print("\n Top pickup locations in last 5 seconds:")
df_with_unix.filter(col("pickup_unix") >= (max_unix_time - 5)) \
    .groupBy("PULocationID") \
    .agg(s("passenger_count").alias("recentPassengers")) \
    .orderBy(col("recentPassengers").desc()) \
    .show()
print("\n Top pickup locations in last 10 seconds:")
df_with_unix.filter(col("pickup_unix") >= (max_unix_time - 10)) \
    .groupBy("PULocationID") \
    .agg(s("passenger_count").alias("recentPassengers")) \
    .orderBy(col("recentPassengers").desc()) \
    .show()
```

```
▸ (7) Spark Jobs
▸ df_with_unix:  pyspark.sql.dataframe.DataFrame = [VendorID: long, tpep_pickup_datetime: timestamp_ntz ... 20 more fields]

 Top pickup locations in last 5 seconds:
+------------+----------------+
|PULocationID|recentPassengers|
+------------+----------------+
|          48|               2|
+------------+----------------+


 Top pickup locations in last 10 seconds:
+------------+----------------+
|PULocationID|recentPassengers|
+------------+----------------+
|          48|               2|
+------------+----------------+
```

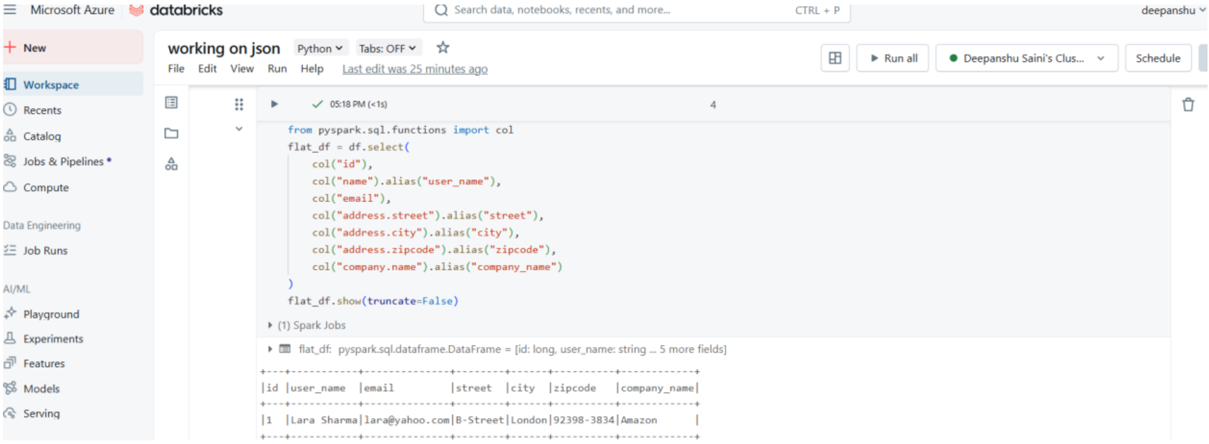
Load any dataset into DBFS



```
df = spark.read.format("json").option("multiLine", "true").load("dbfs:/FileStore/shared_uploads/deepanshusaini2402@gmail.com/week8.json")
```

```
(1) Spark Jobs
df: pyspark.sql.dataframe.DataFrame
    address: struct
        city: string
        street: string
        zipcode: string
    company: struct
        name: string
    email: string
    id: long
    name: string
```

Flatten JSON fields

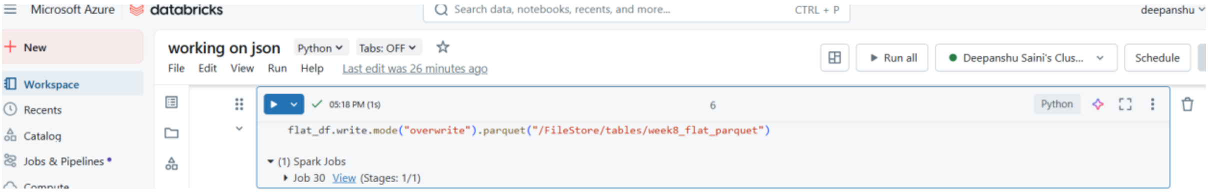


```
from pyspark.sql.functions import col
flat_df = df.select(
    col("id"),
    col("name").alias("user_name"),
    col("email"),
    col("address.street").alias("street"),
    col("address.city").alias("city"),
    col("address.zipcode").alias("zipcode"),
    col("company.name").alias("company_name")
)
flat_df.show(truncate=False)
```

```
(1) Spark Jobs
flat_df: pyspark.sql.dataframe.DataFrame = [id: long, user_name: string ... 5 more fields]

+---+-----------+--------------+--------+------+----------+------------+
|id |user_name  |email         |street  |city  |zipcode   |company_name|
+---+-----------+--------------+--------+------+----------+------------+
|1  |Lara Sharma|lara@yahoo.com|B-Street|London|92398-3834|Amazon      |
+---+-----------+--------------+--------+------+----------+------------+
```

Write flattened file as external parquet table



```
flat_df.write.mode("overwrite").parquet("/FileStore/tables/week8_flat_parquet")
```

```
(1) Spark Jobs
    Job 30  View (Stages: 1/1)
```